COLUMN

シリコンバレーに学ぶキャリア・デベロップメント

═ 第16回(最終回) ═

体力(スタミナ)

東原朋成

日本でも24時間営業のフィットネス・クラブというのが登場してきていると聞きます．シリコンバレーのほとんどの町には，24時間営業のフィットネス・クラブがあります．米国は車社会ですから，電車やバスの時間(始発や最終)に左右されることがないということもありますが，早朝や深夜にもかかわらず，フィットネス・クラブの利用者がいるようです．筆者は通常朝6時に家を出ますが，筆者の住まいの近くにあるフィットネス・クラブに向かう車を目にしないことはありません．筆者の同僚たちの中には朝5時ごろにはもうクラブに行って汗を流している者が何人もいます．ひと汗かいてから出社というわけです．また，昼食時のフィットネス・クラブも非常に混雑していると聞きます．

体力はやはり重要です．某大手企業で部門長をしている筆者の元上司(50歳代後半)は，今でも週3回フィットネス・クラブに通い，週末はクォータ・マラソン(フルマラソンの1/4の距離＝10km強を走る)を自分自身に課しているそうです．彼はゴルフもたしなみますから，それも考慮するとかなりの運動量です．そこまでするのは，健康のためといった次元ではなく，しごとに必要なスタミナが毎年増大しているのに自分を対応させるためだと言います．商談をまとめるために，1週間で世界を1周するようなビジネス旅行に出ることもあります．さらに週末返上でビジネス・プランを立てたり，ビジネス・ミーティングをこなすこともしばしばです．こうした過酷なスケジュールをこなすには，体力(スタミナ)第一というわけです．

また，筆者がかかわったことがある会社のCEO(chief executive officer)は，陸軍士官学校を卒業した退役将校です．士官学校ではリーダシップなどを鍛え上げられるため，米国には士官学校出身の重役が多数いるのですが，彼の体力にはいつも驚かされます．タフガイとはまさに彼のような人物を指すのかと思えるほどです．高年齢ですが，今でも年に1度は軍事教練に参加すると言います．

彼の例は特別なケースになるのかもしれません．しかし，しごとをこなせる体力，しごと量を増やせる体力を日ごろから心がけている人に出会うことはしばしばです．趣味や健康のためという次元を超え，より多くのしごと，責任を果たすことができるための体力を得たいという欲求に駆られてフィットネス・クラブに通う，そうした面々と出会うと，ワーカ・ホリックな米国人の側面をかいま見ることができます．

インタビュー模擬体験

答 前回の解答 (2006年5月号，pp.148-149)

問題

以下に示すCコードを見て，コンパイラの立場からいろいろと批評を加えてください．

```
char a;
for(a=0; a< 20; a++) {
    Some codes
}
```

高級言語(C言語など)では，いくつものデータ・タイプが定義されています．しかし，データ・タイプを真剣に検討するということはあまりないように思います．

組み込みシステムの開発などでは，メモリ容量に制約があるので`short int`や`char`など，メモリを節約できそうなデータ・タイプを適宜選択します．しかし時として，これが行き過ぎることもあります．

例えば`for`におけるループ変数として，ループする回数が8回なので`char`型を使ったとします．一見問題はなさそうなのですが，コンパイラはC言語に忠実でなくてはならないため，不必要な命令を追加することがあります．追加する命令はプロセッサ・アーキテクチャなどによっても異なりますが，この場合など`char`，`int`で実行に変化がないはず(ないとは言い切れないが)ですから，`int`を選択するにこしたことはありません．

高級言語でプログラムを書いてみて，思ったとおりの性能が得られないという状況によく陥ります．これまでに扱ったことがないプロセッサを相手にする場合には，これまでのプログラミング・テクニック，コンパイラ・フラグが思ったように働いてくれないという状況に陥るのは茶飯事かもしれません．

コンパイラには，プロセッサの命令セットやレジスタの数などからくるさまざまな“くせ”があるものです．そうしたくせを知ることは，組み込みプログラミングでは非常に重要です．そのくせを知るためには，簡単なプログラムを書いてコンパイルし，結果を解析するという地道な作業が必要になります．

一方，上述したようにC，C++の言語体系からくる制約など，プラットホームに依存しない事項もたくさんあります．C言語関連の入門書ではこういったことがあまり触れられていませんが，効率の良いプログラムを設計するのにはとても重要になります．

ポインタにまつわるプログラミングのコツや，エンジニアが「こうすればいいのでは？」と思ってやったことがかえってアダになるということもあります．そういったさまざまなことをある程度体系的に調べてみるのは，スキル・アップ（ひいてはキャリア・アップ）に欠かせない重要事項です．

ソフトウェア開発において，オブジェクト指向プログラミングやモジュール化といった手法が注目されています．ソース・コードを将来に渡って活用・保守するためには重要な課題です．アセンブリ言語を使って書かれたプログラムを将来保守しなければならないと考えると，恐ろしくなることもあります．

一方，限られた資源（メモリ容量やプロセッサの演算能力など）で，機能（扱うデータ量など）を向上し続けることをソフトウェア・エンジニアは求められています．

しかし，筆者がこれまで吟味してきたソース・コードには，かなりのむだが散見されました．不必要なスタック・アクセスが頻発するようなソース・コードになっていたり，不必要なモジュール化がなされているのを発見することもしばしばです．

前回の問題はその一つになります．こういったことを日ごろから研究しておくことは，オブジェクト指向のプログラミング手法を研究することと同じくらい重要な要素です．ひょっとすると組み込み系のプログラミングでは，こちらのほうがより重要なのかもしれません．

もちろんツールのサポートはあります．モジュール化されたソース・コードを自動的にインライン化するオプションを備えたコンパイラもあります．効率の良いライブラリ（ターゲットとするプロセッサ向けに最適化されている）を提供するベンダもあります．

コンパイラ・オプションを試してみて，そのくせ（ソース・コードとの相性など）を研究することや，ライブラリ・ルーチンのアセンブリ・コードを読んでみることもたいせつです．実際，同僚のソフトウェア・エキスパートは，日ごろからこうした研究に取り組んでいるようです．彼らは大学でオブジェクト指向プログラミングやモジュール化などの手法の勉強とトレーニングを積んできました．それを現場で日ごろから研究することで，第一線のエンジニアとして活躍し続けることができるのです．

＊　　＊　　＊

これまで16回にわたり，シリコンバレー流のキャリア・アップの側面を紹介してきましたが，今回で最終回とさせていただきます．したがって，問題は用意しません．

今後機会をあらためて，シリコンバレーのエンジニアを素材にしたコラムをお届けしたいと考えています．そのときには，インタビュー模擬体験も再開したいと思います．

とうはら・ともなり